EXHIBIT 26

UNITED STATES DISTRICT COURT
NORTHERN DISTRICT OF CALIFORNIA
SAN JOSE DIVISION

CISCO SYSTEMS, INC., Plaintiff, v. ARISTA NETWORKS, INC., Defendant.	Case No. 5:14-cv-05344-BLF (PSG)

OPENING EXPERT REPORT OF KEVIN JEFFAY, PH.D. REGARDING INFRINGEMENT OF U.S. PATENT NO. 7,047,526

June 3, 2016

HIGHLY CONFIDENTIAL – ATTORNEYS' EYES ONLY – SOURCE CODE

of whatever type that child is. For example, suppose that this subrule is a TokenRule. TokenRule.inhale [CliRule.py : 496-540] uses a matcher (used on line [CliRule.py : 514] and initialized within Rule's constructor [CliRule.py : 487]) to match the input "token" to the rule token.

(c) Update 'context' state

106. If the TokenRule successfully matches the input token to the rule token, then it updates its context state (from NoMatch to the matched value) and invokes its value function (if defined). The ConcatRule thereby removes this TokenRule from its context's list of subrules that remain to be matched. Finally, the OrRule keeps the ConcatRule in its context's list of subrules that could match.

107. If, the TokenRule does not successfully match, then it returns NoMatch, which ConcatRule will return in its inhale() function. The OrRule thus removes that ConcatRule from its context state's subrules. Specifically, on lines [CliRule.py : 1028-1033], tryToInhale() will add this ConcatRule to a list of "refusers." On line [CliRule.py : 1140] of inhale(), endChild() is called on each child in the list of refusers, and removes child from the context.rulemap_.

(d) Return from tryToInhale()

108. Finally, OrRule.tryToInhale() will return with NoMatch if there are no subrules that accept all of the tokens.

109. Picking up after OrRule.tryToInhale() (*i.e.*, in parseAndExecute [CliParser.py : 264-307], a method in the Mode class), after the function has inhaled all of the user-supplied tokens (lines [CliParser.py : 286-288]), it calls instanceRule.inhale(None) [CliParser.py : 292].

110. Like the tokens above, "None" is "percolated" through the OrRule (lines [CliRule.py : 1122-1128]) and the ConcatRule (lines [CliRule.py : 869-871]) and through the

TokenRule (lines [CliRule.py : 497-499]), but only through the list of subrules that remain in these rules' context states.

111. This "percolating" of "None" through the parse tree indicates that there are no more tokens left. ConcatRule successfully inhales a 'None' if its list of remaining subrules is empty (meaning that it has matched all of its tokens). If this happens, then it invokes its value function as described next.

6. **Invoking value functions**

112. Every class that inherits from Rule calls invokeValueFunction() after successfully "inhaling" a token.

- TokenRule calls invokeValueFunction() in inhale() on line [CliRule.py : 531]
- ConcatRule calls invokeValueFunction() in inhale() on line [CliRule.py : 898]
- OrRule calls invokeValueFunction() in inhale() on line [CliRule.py : 1134]
- MultiRangeRule calls invokeValueFunction() in inhale() on line [MultiRangeRule.py : 743]

113. Additionally:

- SetRule inherits from OptionalRule on line [SetRule.py : 90], and defines inhale() to explicitly call OptionalRule.inhale()
- QuotedStringRule [StringRule.py : 87] has an inhale() function that calls its value() function that calls invokeValueFunction().

114. Recall that, when adding commands with addCommand(), there is typically a single callback function ("value function") provided. For instance, recall the example from [.../src/stp/CLIPlugin/StpCli.py : 60]:

configMode.addCommand((tokenStp, tokenForwardTime, forwardTime, setForwardTime))

115. In this example, setForwardTime is the value function. Recall that the rule() function would take this tuple and create a ConcatRule out of it, and set that ConcatRule's value function to setForwardTime(). Thus, when the ConcatRule successfully inhales the final token ('eol'), then it will call setForwardTime().

116. Each of the other rules (tokenStp, tokenForwardTime, and forwardTime) also have a value function, but in this particular example, they would are set to 'None', and thus would not be invoked during parsing. However, the mechanisms are in place that it would be possible to set a value function for each of these tokens.

7. **Sysdb and Agents**

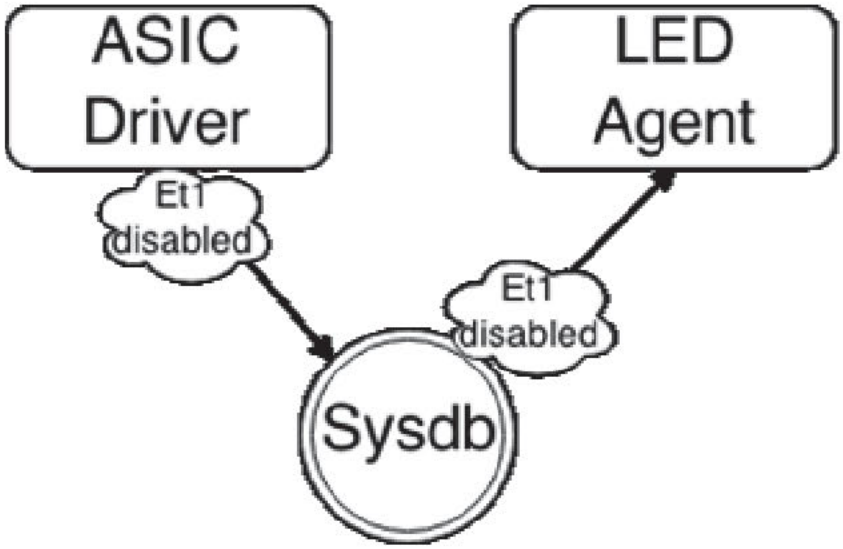
117. According to Arista documentation, Sysdb can be described as follows:

> Enter Sysdb, a small in-memory system that's somewhat like a database, or a file system, or a message bus, depending on how you see it. All the state that needs to be acted on by agents is stored in Sysdb, and each agent materializes a view of a subset of Sysdb that is relevant to its activities. For instance, the LED Driver cares about the status of the ports, but doesn't care about the routing table, therefore it loads only the portion of the state hierarchy that contains the interface statuses.

Arista GitHub: "Understanding EOS and Sysdb," available at https://github.com/aristanetworks/EosSdk/wiki/Understanding-EOS-and-Sysdb

118. The Arista documentation continues:

> The goal of Sysdb is to synchronize state across the different agents, so that the LED agent can receive a notification when ASIC driver marks an interface as "down". This process works in the following steps:



1. The ASIC driver notices that, say, Ethernet1's transceiver has been removed.

2. It then updates its local state hierarchy, setting interface/status/Ethernet1/oper_status tonot connected

3. The underlying agent infrastructure enqueues a state update message on its connection to Sysdb.

4. Sysdb reads the state update off of its end of the connection and updates its state hierarchy appropriately. At this point, both Sysdb and the ASIC driver know that Ethernet1 is not connected

5. Sysdb sends this state update to every other agent who has registered interest in that state update.

6. The LED agent, which registered interest in interface state at start-up, reads the update off of its connection to Sysdb and updates its copy of the interface state hierarchy.

7. This triggers a callback so the LED agent can switch off the little light, indicating that signal onEthernet1 was lost.

Arista GitHub: "Understanding EOS and Sysdb," available at https://github.com/aristanetworks/EosSdk/wiki/Understanding-EOS-and-Sysdb

119. When a CLI command creates a change in state by, for example, changing a configuration, state update notifications are propagated across the system, including to the agents

I certify under penalty of perjury that the foregoing is true and correct.

Date: June 3, 2015.

Kevin Jeffay, Ph.D.